



15370 Barranca Parkway
Irvine, CA 92618-2215



RFID Reader DLL

USER MANUAL

© 2008 HID Global Corporation. All rights reserved.

4503-USM-01-0-02

Software Version 4.1

November 4, 2008

Doc Number: 4503-USM-01, Rev A.0

Warning - Read before start-up!

- The product may only be used for the intended purpose designed by for the manufacturer. The operation manual should be conveniently kept available at all times for each user.
- Unauthorized changes that have not been sold or recommended by the manufacturer may have a negative influence on the system the program has been installed or copied on. Such unauthorized measures shall exclude any liability by the manufacturer.
- The liability-prescriptions of the manufacturer in the issue valid at the time of purchase are valid for the device. The manufacturer shall not be held legally responsible for inaccuracies, errors, or omissions in the manual or automatically set parameters for a device or for an incorrect application of a device.
- Only qualified personnel should carry out installation, operation, and maintenance procedures.
- Use of the program and its installation must be in accordance with national legal requirements.
- When working on devices the valid safety regulations must be observed.
- International copyrights are applicable to this program. Unauthorized copying, distribution or resale of this program or of parts of this program is a violation of applicable laws and will be prosecuted.

Read Me First

About This Guide

This manual describes the Reader DLL functionality. Its goal is to describe the software package, how it works, how to integrate it and how to use it.

Contacts

Europe, Middle East and Africa
HID Global Corporation, Ltd. (Haverhill, UK) email: eusupport@hidglobal.com main: +44 (0) 1440 714 850 support: +44 (0) 1440 711 822 fax: +44 (0) 1440 714 840

Contents

Warning - Read before start-up!	2
Read Me First	3
Scope	5
Overview	5
1 Commands	7
2 DESFire command set	42
3 Script language	65
4 References	89
APPENDIX A - C++ Example Listing.....	90
APPENDIX B - C++ Sample Application.....	91
APPENDIX C - VB .NET without Marshal Casting	92
APPENDIX D - Static Library	93
APPENDIX E - Paypass functionality	93
APPENDIX F - Firmware update.....	93
APPENDIX G - Version History	94

Scope

The Reader DLL is a basic library to support an easy communication with different reader types. Most significant features are:

- Supports various reader types.
- Script language for an easy standard communication with different readers.
- DESFire command set with SAM support.

This documentation is divided into three main parts:

- Explanation of device and reader communication
- Explanation of the script language.
- Appendix with examples in C++, flow diagram, etc.

Overview

Definitions

Source Code

Source code is printed in the following way:

```
printf("Source code is displayed like this\n");
```

Zero terminated string

Zero terminated strings are strings containing a trailing zero byte

hexadecimal dump of string	String
54 65 73 74 00	'Test'

ASCII notation

Each byte is represented as a two character hexadecimal number.

Command	Parameter	Description
read register	0A	Read register 0Ah. Data has 2 characters/bytes.

Binary notation

Each byte is represented as one byte.

Command	Parameter	Description
read register	0Ah	Read register 0Ah. Data has 1 character/byte.

Abbreviations

Abbreviation	Description
ASCII	American Standard Code for Information Interchange
BIN	binary
VB	Visual Basic
VB6	Visual Basic 6.0
VB7	Visual Basic .NET
ABh	This represents a hexadecimal number AB. C++: 0xAB VB: &HAB
<CR>	Carriage return (0Dh).
<LF>	Line feed (0Ah).
CRC	cyclic redundancy checking
broadcast	Broadcast is only defined in binary mode. In broadcast mode station id FFh is used to send a command.

Supported Devices

Reader

- Mifare 1.0
- all Dual 2.x
- all Bootloader 0.x and 1.x
- This device only supports binary protocol.
- all LFX 1.x
- all MultiISO 1.x

Host Systems

- Windows 2000/XP
- Windows CE: Pocket PC 2002/2003

Only one device at the same time is supported.

1 Commands

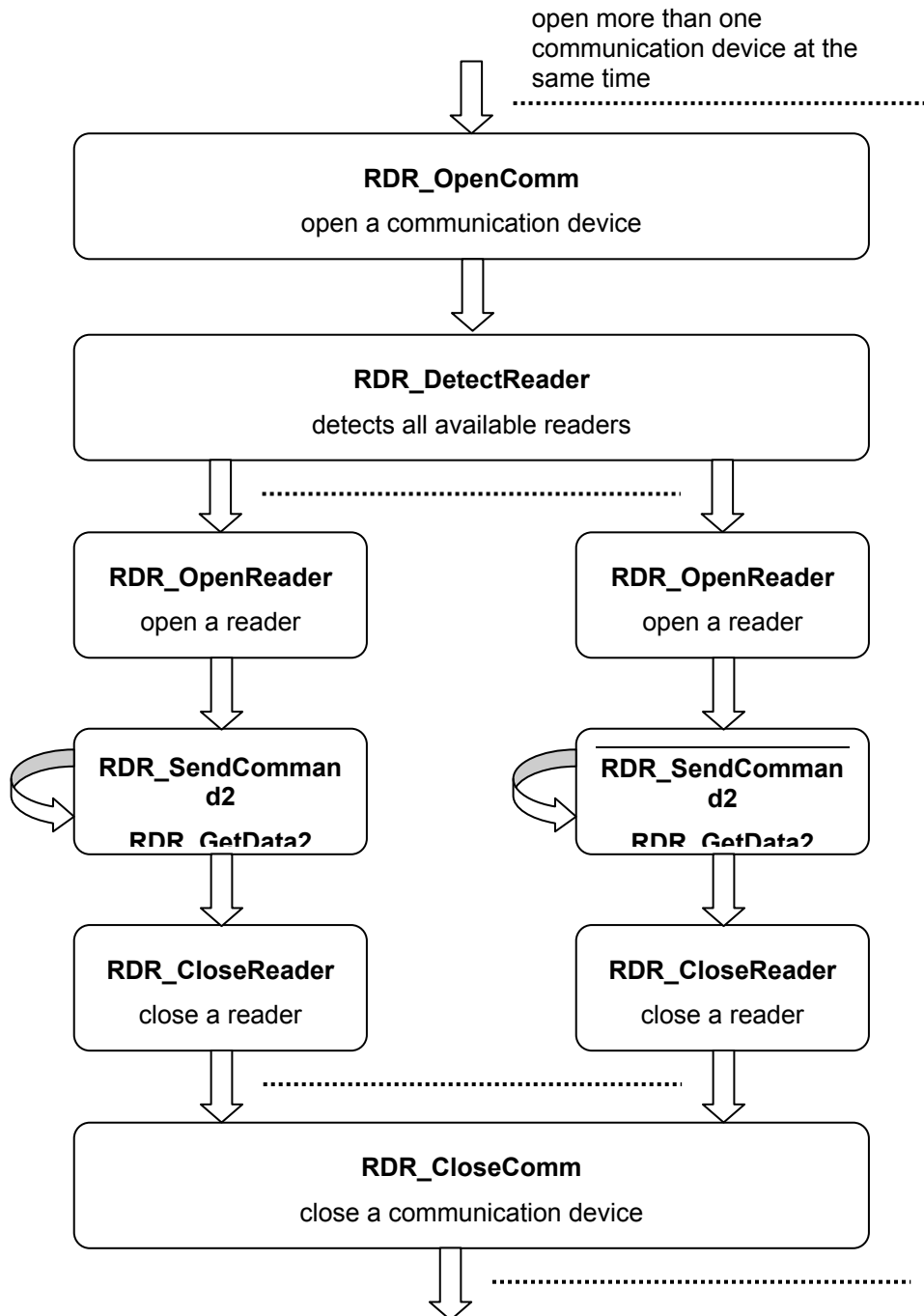
1.1 Function

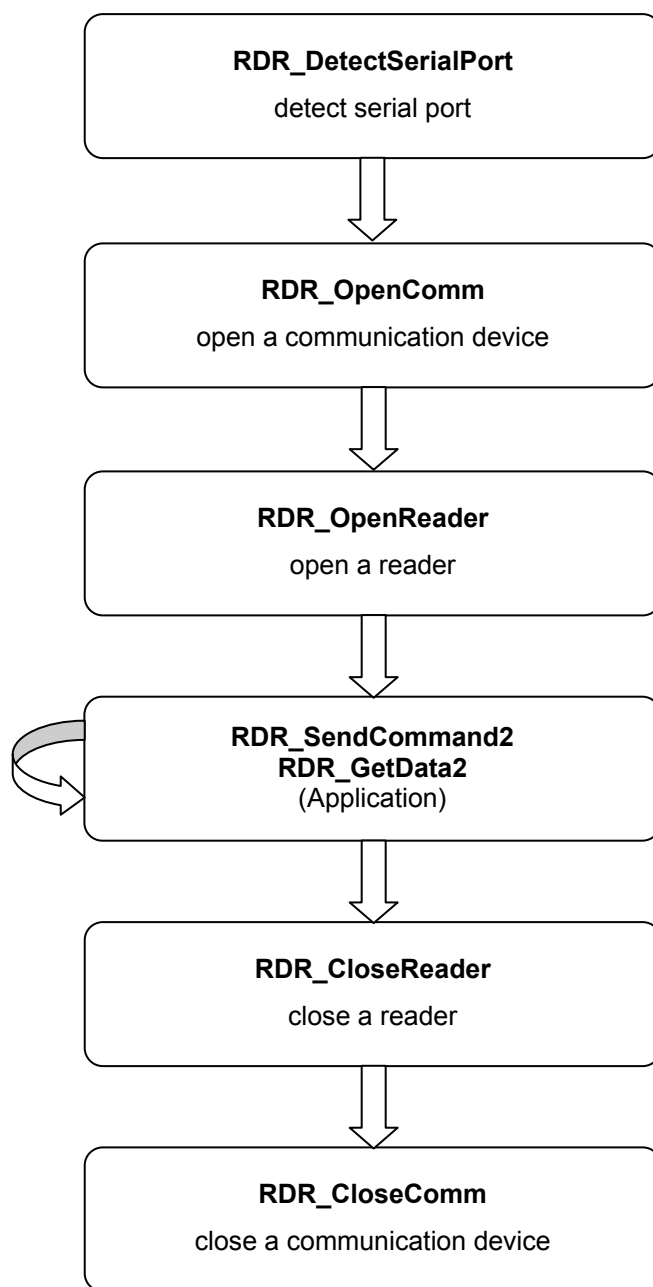
Command	Description
Device communication	
RDR_AbortContinuousRead	Abort continuous read command in ASCII mode
RDR_AbortContinuousReadExt	Abort continuous read command in ASCII and binary mode
RDR_CloseComm	Close communication device
RDR_CommExtFunction	Gives control for the RTS and DTR pin
RDR_DetectSerialPort	Detect the serial port of a connected reader
RDR_EmptyCommRcvBuffer	Empty the receive buffer of the communication device
RDR_GetCommBaudRate	Get baud rate
RDR_GetCommContRcv	Get state of continuous receive mode
RDR_GetCommDeviceHandle	Get communication device handle
RDR_GetCommProtocol	Get protocol type
RDR_GetCommTimeout	Get the timeout of the communication device
RDR_OpenComm	Open communication device
RDR_OpenComm_OxfordBoard	Open communication device for Oxford chipset
RDR_OpenSingle	Opens a communication device and a reader
RDR_SetCommBaudRate	Set baud rate
RDR_SetCommBaudRate_OxfordBoard	Set baud rate for Oxford chipset
RDR_SetCommContRcv	Set continuous receive mode of the communication device
RDR_SetCommProtocol	Set protocol type (ASCII / BIN)
RDR_SetCommTimeout	Set the timeout of the communication device
RDR_SleepComm	Sleep communication device
RDR_WakeupComm	Wake up communication device
Reader communication	
RDR_CloseReader	Close a reader
RDR_DetectReader	Detect available readers
RDR_GetBinFlag	Get Flag of binary protocol v2
RDR_GetBinProtocol	Get used binary protocol version
RDR_GetBroadcast	Get state of broadcast
<i>RDR_GetData</i> (obsolete, use RDR_GetData2)	Get data from a reader
RDR_GetData2	Get data from a reader
<i>RDR_GetDataTimeout</i> (obsolete, use RDR_GetDataTimeout2)	Get data from a reader with timeout

Command	Description
RDR_GetDataTimeout2	Get data from a reader with timeout
RDR_GetDebugOutput	Get debug output
RDR_GetDebugOutputState	Get state of debug output
RDR_GetDeviceID	Get device ID of a reader
RDR_GetReaderConfig	Get reader configuration
RDR_GetReaderType	Get version string of a reader
RDR_GetStationID	Get the station id of a reader
RDR_GetTiming	Get response timing
RDR_InitUpdateFirmware	Initialize the firmware update
RDR_IsCommandAvailable	Check if script command is available for this reader
RDR_LoadFirmware	Prepares the firmware upload and checks the password
RDR_OpenReader	Opens a reader
RDR_ResetReader	Reset a reader
RDR_SendCommand (obsolete, use RDR_SendCommand2)	Send a command to a reader
RDR_SendCommand2	Send a command to a reader
RDR_SendCommandGetData (obsolete, use RDR_SendCommandGetData2)	Send a command to a reader and get data
RDR_SendCommandGetData 2	Send a command to a reader and get data
RDR_SendCommandGetData Timeout (obsolete, use RDR_SendCommandGetDataTimeout2)	Send a command to a reader and get data with timeout
RDR_SendCommandGetData Timeout2	Send a command to a reader and get data with timeout
RDR_SetBroadcast	Set broadcast on or off in binary mode
RDR_SetDebugOutputState	Set state of debug output
RDR_SetReaderConfig	Set reader configuration
RDR_StartTimer	Start response timer
RDR_UpdateFirmware	Update one line of the firmware.
Windows CE specific functions	
RDR_GetHighBaudrates	Get activation status of higher baud rates
RDR_GetResumeState	Get the power resume state
RDR_SetHighBaudrates	Activates/Deactivates the higher baud rates
Other functions	
RDR_DESDecrypt	DES Decryption
RDR_DESEncrypt	DES Encryption
RDR_DESFire	DESFire Tag and SAM command set
RDR_GetDLLVersion (obsolete, use RDR_GetDLLVersionStr)	Get version of DLL

Command	Description
RDR_GetDESFireSAMTimeout	Get DESFire SAM Timeout
RDR_GetDLLVersionStr	Get version of DLL as string
RDR_SetDESFireSAMTimeout	Set DESFire SAM Timeout

1.2 Windows 2000/XP





1.2.1 RDR_AbortContinuousRead

The **RDR_AbortContinuousRead** function aborts the continuous read command in ASCII mode and empties the buffer.

Win **RDR_AbortContinuousRead (void* hComm);**

CE **RDR_AbortContinuousRead ();**

Parameter	Description
hComm	Handle of communication device to abort continuous read command.

1.2.2 RDR_AbortContinuousReadExt

The **RDR_AbortContinuousReadExt** function aborts the continuous read command in ASCII and binary mode and empties the buffer.

Win **RDR_AbortContinuousReadExt (void* hReader);**

CE **RDR_AbortContinuousReadExt ();**

Parameter	Description
hReader	Handle of reader to abort continuous read command.

1.2.3 RDR_CloseComm

The **RDR_CloseComm** function closes and releases the communication device.

Win **RDR_CloseComm(void* hComm);**

CE **RDR_CloseComm();**

Parameter	Description
hComm	Handle of communication device to close.

1.2.4 RDR_CommExtFunction

The **RDR_CommExtFunction** function gives control of the RTS and DTR pin.

Win **RDR_CommExtFunction(void* hComm, long function);**

CE **RDR_CommExtFunction(long function);**

Parameter	Description
hComm	Handle of communication device to control
function	Function you want to control The following values are valid and defined in windows.h or winbase.h: CLRDTT CLRRTS SETDTT SETRTS

1.2.5 RDR_DetectSerialPort

The **RDR_DetectSerialPort** function detects the serial ports of connected reader.

Win **RDR_DetectSerialPort (void* hComm, char* buffer);**

CE **RDR_DetectSerialPort (char* buffer);**

Parameter	Description
hComm	Handle of communication device to control
buffer	A buffer of the return value. The buffer size has the range of 10 bytes for CE version and 256 bytes for the normal version.

Return value

All found serial ports with terminating 0 are returned. The answer should be parsed until a 0 length string was found.

Note that only one string will be returned using CE version, so there is no need to parse the answer.

Example of a detected serial port: "COM4:"

1.2.6 RDR_EmptyCommRcvBuffer

The **RDR_EmptyCommRcvBuffer** function empties the receive buffer of the communication device.

Win **RDR_EmptyCommRcvBuffer(void* hComm);**
CE **RDR_EmptyCommRcvBuffer();**

Parameter	Description
hComm	Handle of communication device to empty receive buffer.

1.2.7 RDR_GetCommBaudRate

The **RDR_GetCommBaudRate** function gets the current baud rate of the communication device.

Win **long RDR_GetCommBaudRate(void* hComm);**
CE **long RDR_GetCommBaudRate();**

Parameter	Description
hComm	Handle of communication device to get baud rate.

Return value

The actually used baud rate of the communication device is returned.

1.2.8 RDR_GetCommContrcv

The **RDR_GetCommContrcv** function returns the state of continuous receive mode.

Win **char RDR_GetCommContrcv(void* hComm);**
CE **char RDR_GetCommContrcv();**

Parameter	Description
hComm	Handle of communication device to get continuous receive mode.

Return value

If the continuous receive mode is active, the return value is one or two.

If the continuous receive mode is not active, the return value is zero.

Remarks

For more detailed information about continuous receive mode refer to **RDR_SetCommContrcv**.

1.2.9 RDR_GetCommDeviceHandle

The **RDR_GetCommContrCv** function returns the communication device handle.

Win **void*** RDR_GetCommDeviceHandle(void* hComm);

CE **void*** RDR_GetCommDeviceHandle();

Parameter	Description
hComm	Handle of communication device to get continuous receive mode.

Return value

The communication device handle is returned. If using a serial port as communication device the file handle is returned.

1.2.10 RDR_GetCommProtocol

The **RDR_GetCommProtocol** function returns the active transfer protocol of the communication device.

Win **char** RDR_GetCommProtocol(void* hComm);

CE **char** RDR_GetCommProtocol();

Parameter	Description
hComm	Handle of communication device to get protocol.

Return value

If the ASCII protocol is used, the return value is zero.

If the BIN protocol is used, the return value is one.

1.2.11 RDR_GetCommTimeout

The **RDR_GetCommTimeout** function returns the timeout of the communication device within the library.

Win **long** RDR_GetCommTimeout(void* hComm);

CE **long** RDR_GetCommTimeout();

Parameter	Description
hComm	Handle of communication device to get timeout.

Return value

The timeout in milliseconds is returned.

Remarks

If the communication device handle is zero, the returned timeout value is the global timeout. For more information about global timeout refer to **RDR_SetCommTimeout**.

1.2.12 RDR_OpenComm

The **RDR_OpenComm** function initializes and opens a specified communication device.

Win `void* RDR_OpenComm(char* device, char autodetect, struct presetSettings* settings);`

CE `char RDR_OpenComm(char* device, char autodetect, struct presetSettings* settings);`

Structure Definition

```
struct presetSettings {
    long baudRate;
    char protocol; };
```

Parameter	Description
device	<p>0 terminated string containing the communication device. "com1", "com2", or any other string terminated with 'com' number. For example, abc1 to abc200.</p> <p>The device buffer size has the range of 128 bytes.</p>
autodetect	<p>0 ... The auto detection of the baud rate and the protocol is deactivated. Values from <i>settings</i> are used.</p> <p>1 ... The normal auto detection of the baud rate and the protocol is activated.</p> <p>2 ... The fast auto detection of the baud rate and the protocol is activated. Older reader devices may not be found.</p> <p>4 ... If this bit is additionally set, not common baud rates are disabled (19200, 38400, 230400). Use this to speed up the autodetect procedure.</p> <p>If MSB (80h) is set the power management for Windows CE will be deactivated.</p>
settings	Pointer to the structure of preset settings.

Windows CE

If the power management is deactivated the user has to manage it.

Return value

If the function succeeds, the return value is a handle of the opened communication device.

If the communication port is opened on a Windows CE device successfully the return value is one.

If the function fails, the return value is zero.

Remarks

Normal auto detection needs several seconds.

If more than one reader is connected to RS485 use this mode.

Fast auto detection only works correctly with newer reader devices and only with one reader connected.

For example the reader "Mifare 0.14e" can not be found with fast auto detection in binary mode.

For unsupported reader types the initialization may fail.

1.2.13 RDR_OpenComm_OxfordBoard

Win `void* RDR_OpenComm_OxfordBoard(char* device, char autodetect, struct presetSettings* settings);`

CE `char RDR_OpenComm_OxfordBoard (char* device, char autodetect, struct presetSettings* settings);`

This specific function can be called to give a 1:1 relationship between the requested baud rate and the actual baud rate. In others words the bdRate passed to the function is actual baud rate on which board will communicate..unlike previous function, where we need to pass 4xbdRate.

This passing baud rate setting will only be usefull when auto detection is set to false.

1.2.14 RDR_OpenSingle

The **RDR_OpenSingle** function opens a communication device and a reader at once.

This function only works for one reader on the communication device.

Win `char RDR_OpenSingle(void** hComm, void ** hReader, char* device, char autodetect, short knownReader, struct presetSettings* settings);`

CE `char RDR_OpenSingle(char* device, char autodetect, short knownReader, struct presetSettings* settings);`

Parameter	Description
hComm	Handle of communication device.
hReader	Handle of reader.
device	0 terminated string containing the communication device. For more detailed information refer to <code>RDR_OpenComm</code> .
autodetect	Auto detection of the baud rate and the protocol. For more detailed information refer to <code>RDR_OpenComm</code> .
knownReader	Auto detect or detect manually reader type. For more detailed information refer to <code>RDR_OpenReader</code> .
settings	Pointer to the structure of preset settings.

Return value

If the function succeeds, the return value is not zero.

If the function fails, the return value is zero.

1.2.15 RDR_SetCommBaudRate

The **RDR_SetCommBaudRate** function sets and applies the baud rate of the communication device.

Win **RDR_SetCommBaudRate(void* hComm, long bdRate);**

CE **RDR_SetCommBaudRate(long bdRate);**

Parameter	Description
hComm	Handle of communication device to set baud rate.
bdRate	Baud rate of communication device. (default: 9600)

Remarks

Valid baud rates are 9600, 19200, 38400, 57600, 115200, 230400 and 460800.

This command does not apply the baud rate of the read module.

Be sure the used interface supports the baud rate.

1.2.16 RDR_SetCommBaudRate_OxfordBoard

The **RDR_SetCommBaudRate_OxfordBoard** function sets and applies the baud rate of the communication device.

Win **RDR_SetCommBaudRate_OxfordBoard(void* hComm, long bdRate);**

CE **RDR_SetCommBaudRate_OxfordBoard(long bdRate);**

Parameter	Description
hComm	Handle of communication device to set baud rate.
bdRate	Baud rate of communication device. (default: 9600)

Remarks

It is similar to **RDR_SetCommBaudRate** except the passing baud rate will be the actual baud rate on which reader board will communicate rather than **4xBaudRate**.

1.2.17 RDR_SetCommContrCv

The **RDR_SetCommContrCv** function sets and applies the continuous receive mode of the communication device.

Win **RDR_SetCommContrCv(void* hComm, char contReceiveMode);**

CE **RDR_SetCommContrCv(char contReceiveMode);**

Parameter	Description
hComm	Handle of communication device to set continuous receive mode.
contReceiveMode	<p>0 ... string mode ASCII: a complete line without <CR> and <LF> is returned from RDR_GetData. BIN: only data of a complete and valid frame without frame overhead is returned from RDR_GetData.</p> <p>The communication timeout is used.</p> <p>1 ... raw mode RDR_GetData returns a buffer with a leading length byte of the buffer and all received bytes (including the frame data in binary mode and <CR> and <LF> in ASCII mode).</p> <p>No communication timeout is used. All RDR_GetData functions are returning immediately.</p> <p>2 ... raw string mode RDR_GetData returns a combination of string mode and raw mode. In ASCII mode multiple strings are separated with <CR> and <LF>, in binary mode only one string is sent with leading length byte.</p> <p>All RDR_GetData functions are returning immediately.</p>

Remarks

Even incomplete or corrupted data is returned from **RDR_GetData** if continuous receive mode is in raw mode.

A complete frame in BIN includes a frame overhead with a CRC value.

For binary protocol refer to reader documentation.

If polling is used to receive data from the reader use mode 1 and 2, because they do not use the communication timeout and return immediately.

1.2.18 RDR_SetCommProtocol

The **RDR_SetCommProtocol** function sets and applies the transfer protocol of the communication device.

Win **RDR_SetCommProtocol**(void* hComm, char protocol);

CE **RDR_SetCommProtocol**(char protocol);

Parameter	Description
hComm	Handle of communication device to set protocol.
protocol	0 ... ASCII protocol (default) 1 ... BIN protocol

Remarks

This command does not apply the protocol of the read module.

1.2.19 RDR_SetCommTimeout

The **RDR_SetCommTimeout** function sets and applies the timeout of the communication device within the library.

Win **RDR_SetCommTimeout**(void* hComm, long timeout);

CE **RDR_SetCommTimeout**(long timeout);

Parameter	Description
hComm	Handle of communication device to set timeout.
timeout	Timeout in milliseconds. (default: 300)

Remarks

If the communication device does not respond within timeout, a communication operation will be canceled.

If timeout exceeded, ensure that the communication is not broken or increase timeout value.

If the communication device handle is zero, the timeout in the library is set globally.

Every new opened communication device uses the globally timeout as default.

Increasing the timeout will be slower the detection of the reader.

1.2.20 RDR_SleepComm

The **RDR_SleepComm** function closes the communication device but does not release the used resources. Use the **RDR_WakeupComm** function to wake up the communication.

This two functions should be used if closing and reopening the communication device takes too much time. This functions are much faster, because they do not make any auto detection.

Win **RDR_SleepComm(void* hComm);**

CE **RDR_SleepComm();**

1.2.21 RDR_WakeupComm

The **RDR_WakeupComm** function reopens the communication device that was previously send to sleep mode. Use the **RDR_SleepComm** function to set the communication device into sleep mode.

This two functions should be used if closing and reopening the communication device takes too much time. This functions are much faster, because they do not make any auto detection.

Win **char RDR_WakeupComm(void* hComm);**

CE **char RDR_WakeupComm();**

Return value

If the function succeed, the return value is one.

If the function fails, the return value is zero.

1.3 Reader communication

1.3.1 RDR_CloseReader

The **RDR_CloseReader** function closes a reader specified with its handle.

Win **RDR_CloseReader(void* hReader);**

CE **RDR_CloseReader();**

Parameter	Description
hReader	Handle of reader to close.

1.3.2 RDR_DetectReader

The **RDR_DetectReader** function detects all available readers and devices on a specified communication device.

Win **char* RDR_DetectReader(void* hComm, char* buffer);**

CE **char RDR_DetectReader();**

Parameter	Description
hComm	Handle of communication device to detect reader.
buffer	A buffer of the return value. The buffer size has the range of 256 bytes.

Return value

A 0 terminated string containing all detected IDs is returned.

Hex-dump example of detected readers with the station IDs 03h, 05h, 80h and FEh:

030580FE00

Windows CE returns only the detected ID from one reader.

Remarks

Auto detection needs several seconds.

This command is only needed in binary mode to get the correct ID.

1.3.3 RDR_GetBinFlag

The **RDR_GetBinFlag** function returns the received flag byte (from binary protocol version 2) from the last read data of a reader specified with its handle.

Win **char RDR_GetBinFlag (void* hReader);**
CE **char RDR_GetBinFlag ();**

Parameter	Description
hReader	Handle of reader to get binary flag byte.

Return value

The flag byte of binary protocol v2 is returned.

Remarks

This function can only be called once. After calling this function the binary flag is set to -1.

1.3.4 RDR_GetBinProtocol

The **RDR_GetBinProtocol** function returns the used binary protocol version of a reader specified with its handle.

Win **char RDR_GetBinProtocol (void* hReader);**
CE **char RDR_GetBinProtocol ();**

Parameter	Description
hReader	Handle of reader to get binary flag byte.

Return value

The version of used binary protocol is returned.

If ASCII protocol is active, zero is returned.

1.3.5 RDR_GetBroadcast

The **RDR_GetBroadcast** function returns the state of broadcast mode of a reader specified with its handle.

Win **char RDR_GetBroadcast(void* hReader);**
CE **char RDR_GetBroadcast();**

Parameter	Description
hReader	Handle of a reader to get the state of broadcast mode.

Return value

If broadcast is deactivated, the return value is zero.

If broadcast is activated, the return value is one.

1.3.6 RDR_GetData (obsolete)

The **RDR_GetData** function receives data from a reader specified with its handle.

Win `char* RDR_GetData(void * hReader, char* buffer);`

CE `char* RDR_GetData(char* buffer);`

Parameter	Description
hReader	Handle of reader to receive data.
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.

Return value

In ASCII protocol mode the function returns a 0 terminated string containing received data.

In binary protocol mode the function returns a buffer of received data including a leading length byte.

Hex-dump example of return value "Dual 2.0" with ASCII protocol:

4475616C20322E3000

Hex-dump example of return value "Dual 2.0" with binary protocol:

084475616C20322E30

Remarks

In continuous receive mode the user has to check data integrity by its own.

For valid data frame format refer to reader documentation.

If **RDR_GetData** is used for polling the continuous receive mode has to be activated.

This function is obsolete, use **RDR_GetData2**.

1.3.7 RDR_GetData2

The **RDR_GetData2** function receives data from a reader specified with its handle.

Win **char* RDR_GetData2(void * hReader, char* buffer, unsigned int *length);**

CE **char* RDR_GetData2(char* buffer, unsigned int *length);**

Parameter	Description
hReader	Handle of reader to receive data.
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.
Length	Length of valid characters in the return buffer.

Return value

In ASCII protocol mode the function returns a 0 terminated string containing received data.

Hex-dump example of return value "Dual 2.0" with ASCII protocol (length = 8):

4475616C20322E3000

Hex-dump example of return value "Dual 2.0" with binary protocol (length = 8):

4475616C20322E30

Remarks

In continuous receive mode the user has to check data integrity by its own.

For valid data frame format refer to reader documentation.

If **RDR_GetData2** is used for polling the continuous receive mode has to be activated.

1.3.8 RDR_GetDataTimeout (obsolete)

The **RDR_GetDataTimeout** function receives data from a reader specified with its handle with timeout.

Win **char* RDR_GetDataTimeout(void * hReader, char* buffer, long timeout);**

CE **char* RDR_GetDataTimeout(char* buffer, long timeout);**

Parameter	Description
hReader	Handle of reader to receive data.
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.
timeout	Timeout in milliseconds. (Minimum: 100ms)

Return value

Refer to **RDR_GetData**.

This function is obsolete, use **RDR_GetDataTimeout2**.

1.3.9 RDR_GetDataTimeout2

The **RDR_GetDataTimeout2** function receives data from a reader specified with its handle with timeout.

Win `char* RDR_GetDataTimeout2(void * hReader, char* buffer, unsigned int *length, long timeout);`

CE `char* RDR_GetDataTimeout2(char* buffer, unsigned int *length, long timeout);`

Parameter	Description
hReader	Handle of reader to receive data.
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.
length	Length of valid characters in the return buffer.
timeout	Timeout in milliseconds. (Minimum: 100ms)

Return value

Refer to **RDR_GetData2**.

1.3.10 RDR_GetDebugOutput

The **RDR_GetDebugOutput** function receives the debug data from a reader specified with its handle.

Win `char* RDR_GetDebugOutput(void * hReader, char* buffer);`

CE `char* RDR_GetDebugOutput(char* buffer);`

Parameter	Description
hReader	Handle of reader to receive data.
buffer	A buffer of the return value. The buffer size has the range of 2048 bytes.

Return value

The function returns a 0 terminated string containing the debug output.

Remarks

Sent data get a ">>" prefix and received data get a "<<" prefix.

A buffer overflow is signaled with a '&' character.

Received data are always represented as HEX values.

Example

This example shows the debug output of a version command in ASCII mode:

">>v<<4D 75 6C 74 69 49 53 4F 20 31 2E 30 0D 0A"

1.3.11 RDR_GetDebugOutputState

The **RDR_GetDebugOutputState** function returns the state of debug output of a reader specified with its handle.

Win **char RDR_GetDebugOutputState (void* hReader);**

CE **char RDR_GetDebugOutputState ();**

Parameter	Description
hReader	Handle of a reader to get the state of broadcast mode.

Return value

If debug output is deactivated, the return value is zero.

If debug output is activated, the return value is one.

1.3.12 RDR_GetDeviceID

The **RDR_GetDeviceID** function returns the device ID of a reader specified with its handle. The device ID represents the serial number of the reader.

Win **char* RDR_GetDeviceID(void* hReader, char* buffer);**

CE **char* RDR_GetDeviceID(char* buffer);**

Parameter	Description
hReader	Handle of reader to get device ID.
buffer	A buffer of the return value. The buffer size has the range of 20 bytes.

Return value

A 0 terminated string containing the device ID of a reader is returned.

If a reader module does not support the Device ID, the string "N/A" (not available) is returned.

1.3.13 RDR_GetReaderConfig

The **RDR_GetReaderConfig** function returns current reader configuration.

Win **struct readerConfig RDR_GetReaderConfig(void* hReader);**

CE **struct readerConfig RDR_GetReaderConfig();**

Parameter	Description
hReader	Handle of reader to get configuration.

Structure Definition

```

struct readerConfig {
    long baudRate;
    char protocol;
    unsigned char stationID; };

```

For more detailed information refer to **RDR_SetReaderConfig**.

Return value

The function returns a structure of configuration data of a reader.

1.3.14 RDR_GetReaderType

The **RDR_GetReaderType** function returns the version string of a reader specified with its handle.

Win **char*** RDR_GetReaderType(void* hReader, char* buffer);

CE **char*** RDR_GetReaderType(char* buffer);

Parameter	Description
hReader	Handle of reader to get reader type.
buffer	A buffer of the return value. The buffer size has the range of 100 bytes.

Return value

A 0 terminated string containing the version of a reader is returned.

1.3.15 RDR_GetStationID

The **RDR_GetStationID** function returns the station ID of a reader specified with its handle.

Win **unsigned char** RDR_GetStationID(void* hReader);

CE **unsigned char** RDR_GetStationID();

Parameter	Description
hReader	Handle of reader to get station ID.

Return value

The station ID of a reader is returned.

1.3.16 RDR_GetTiming

The **RDR_GetTiming** function returns the response timing of a reader command.

Win **float** RDR_GetTiming (void* hReader);

CE **float** RDR_GetTiming ();

Parameter	Description
hReader	Handle of a reader to get the state of broadcast mode.

Return value

Response timing in milliseconds.

Remarks

The timing is not an exact timing of the response time, because the computer needs some time to process the measurement.

Only use **RDR_SendCommandGetData** to measure the response timing.

1.3.17 RDR_InitUpdateFirmware

The **RDR_InitUpdateFirmware** function initializes the firmware update.

Call this function after the firmware is loaded and before the update process begins.

Win **void RDR_InitUpdateFirmware ();**

CE **void RDR_InitUpdateFirmware ();**

1.3.18 RDR_IsCommandAvailable

The **RDR_IsCommandAvailable** function checks if a script command is available for a reader specified with its handle.

Win **char RDR_IsCommandAvailable(void* hReader, char* command);**

CE **char IsCommandAvailable(char* command);**

Parameter	Description
hReader	Handle of reader to check script command.
command	Script command as 0 terminated string.

Return value

If a script command is supported for the reader, the return value is one.

If a script command is not available for the reader, the return value is zero.

1.3.19 RDR_LoadFirmware

The **RDR_LoadFirmware** function initializes the pointer to the data and checks the password.

Win **long RDR_LoadFirmware(void* hReader, char* data, long length, char* password, char option);**

CE **long RDR_LoadFirmware(char* data, long length, char* password, char option);**

Parameter	Description
hReader	Handle of reader to check script command.
data	Pointer to the firmware data
length	Length of the firmware data
password	Buffer of 16 bytes with password
options	Bit 0 (01h) enables update of EEPROM

Return value

If loading succeeds, the return value represents the number of lines to be updated.

Each line can be updated with the **RDR_UploadFirmware** function.

If the function fails, the return value is below zero.

Error Code	Description
-1	Error: Invalid reader handle
-2	Error: Password not correct or wrong firmware file

1.3.20 RDR_OpenReader

The **RDR_OpenReader** function opens a reader on specified communication device with specified ID.

Win **void* RDR_OpenReader(void* hComm, unsigned char id, short knownReader);**

CE **void* RDR_OpenReader(unsigned char id, short knownReader);**

Parameter	Description
hComm	Handle of communication device to open reader.
id	Valid station ID of the reader.
knownReader	0 ... auto detect reader type
	1 ... unknown reader type
	2 ... Mifare 0.14e / f
	3 ... Mifare 0.15d
	4 ... Mifare 1.0
	6 ... ISO 0.9v
	8 ... MULTITAG 0.10b
	10 ... MULTITAG 0.12a
	11 ... MULTITAG 0.12b
	12 ... Mifare 0.15e
	15 ... all Bootloader 0.x / 1.x
	16 ... all Dual 2.x
	18 ... all MULTITAG 1.x
	19 ... all LFX 1.x
	20 ... all MultiISO 1.x

Return value

If the function succeeds, the return value is a handle of the opened reader.

If the function fails, the return value is zero.

Remarks

If **RDR_OpenReader** cannot find any supported reader type with auto detection, the unknown reader type is used.

The unknown reader type does not support any script command.

In ASCII mode the station ID is ignored.

1.3.21 RDR_ResetReader

The **RDR_ResetReader** function resets a reader specified with its handle.

Win **RDR_ResetReader(void* hReader);**

CE **RDR_ResetReader();**

Parameter	Description
hReader	Handle of reader to reset.

Remarks

After reset the reader starts with its EEPROM settings.

1.3.22 RDR_SendCommand (obsolete)

The **RDR_SendCommand** function sends a command to a reader specified by its handle.

Win **long RDR_SendCommand(void* hReader, char* command, char* data);**

CE **long RDR_SendCommand(char* command, char* data);**

Parameter	Description
hReader	Handle of reader to send a command.
command	0 terminated string with the command.
data	ASCII: 0 terminated string containing the data BIN: length of data (1 byte) + data

Return value

If the function succeeds, the return value is greater than zero.

If the function fails, the return value is zero.

If the command is not available for the reader, the return value is -1.

Remarks

Unsupported commands were sent to the reader without modifications.

This function is obsolete, use **RDR_SendCommand2**.

1.3.23 RDR_SendCommand2

The **RDR_SendCommand2** function sends a command to a reader specified by its handle.

Win `long RDR_SendCommand2(void* hReader, char* command, char* data, unsigned int length);`

CE `long RDR_SendCommand2(char* command, char* data, unsigned int length);`

Parameter	Description
hReader	Handle of reader to send a command.
command	0 terminated string with the command.
data	ASCII: 0 terminated string containing the data BIN: data
length	Valid data length.

Return value

If the function succeeds, the return value is greater than zero.

If the function fails, the return value is zero.

If the command is not available for the reader, the return value is -1.

Remarks

Unsupported commands were sent to the reader without modifications.

1.3.24 RDR_SendCommandGetData (obsolete)

The **RDR_SendCommandGetData** function sends a command to a reader specified by its handle and receives data.

Win `char* RDR_SendCommandGetData(void* hReader, char* command, char* data, char* buffer);`

CE `char* RDR_SendCommandGetData(char* command, char* data, char* buffer);`

Parameter	Description
hReader	Handle of reader to send a command.
command	0 terminated string with the command
data	ASCII: 0 terminated string containing the data BIN: length of data (1 byte) + data
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.

Return value

Refer to **RDR_GetData** with continuous receive mode is off.

This function is obsolete, use **RDR_SendCommandGetData2**.

1.3.25 RDR_SendCommandGetData2

The **RDR_SendCommandGetData2** function sends a command to a reader specified by its handle and receives data.

Win **char*** RDR_SendCommandGetData2(void* hReader, char* command, char* data, unsigned int length, char* buffer, unsigned int *bufLength);

CE **char*** RDR_SendCommandGetData2(char* command, char* data, unsigned int length, char* buffer, unsigned int *bufLength);

Parameter	Description
hReader	Handle of reader to send a command.
command	0 terminated string with the command
data	ASCII: 0 terminated string containing the data BIN: data
length	Valid data length.
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.
bufLength	Length of valid characters in the return buffer.

Return value

Refer to **RDR_GetData2** with continuous receive mode is off.

1.3.26 RDR_SendCommandGetDataTimeout (obsolete)

The **RDR_SendCommandGetDataTimeout** function sends a command to a reader specified by its handle and receives data with timeout.

Win **char*** RDR_SendCommandGetDataTimeout(void* hReader, char* command, char* data, char* buffer, long timeout);

CE **char*** RDR_SendCommandGetDataTimeout (char* command, char* data, char* buffer, long timeout);

Parameter	Description
hReader	Handle of reader to send a command.
command	0 terminated string with the command
data	ASCII: 0 terminated string containing the data BIN: length of data (1 byte) + data
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.
timeout	Timeout in milliseconds. (Minimum: 100ms)

Return value

Refer to **RDR_GetData** with continuous receive mode is off.

This function is obsolete, use **RDR_SendCommandGetDataTimeout2**.

1.3.27 RDR_SendCommandGetDataTimeout2

The **RDR_SendCommandGetDataTimeout2** function sends a command to a reader specified by its handle and receives data with timeout.

Win **char* RDR_SendCommandGetDataTimeout2(void* hReader, char* command, char* data, unsigned int length, char* buffer, unsigned int *bufLength, long timeout);**
CE **char* RDR_SendCommandGetDataTimeout2(char* command, char* data, unsigned int length, char* buffer, unsigned int *bufLength, long timeout);**

Parameter	Description
hReader	Handle of reader to send a command.
command	0 terminated string with the command
data	ASCII: 0 terminated string containing the data BIN: data
length	Valid data length.
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.
bufLength	Length of valid characters in the return buffer.
timeout	Timeout in milliseconds. (Minimum: 100ms)

Return value

Refer to **RDR_GetData2** with continuous receive mode is off.

1.3.28 RDR_SetBroadcast

The **RDR_SetBroadcast** function activates/deactivates broadcast mode of binary protocol using a reader specified with its handle.

Broadcast is able to address all readers on the bus system even the Station ID is unknown.

Win **RDR_SetBroadcast(void* hReader, char broadcast);**
CE **RDR_SetBroadcast(char broadcast);**

Parameter	Description
hReader	Handle of a reader to activate/deactivate broadcasting.
broadcast	0 ... deactivate broadcasting 1 ... activate broadcasting.

1.3.29 RDR_SetDebugOutputState

The **RDR_SetDebugOutputState** function activates/deactivates debug output of a reader specified with its handle.

Win **RDR_SetDebugOutputState (void* hReader, char state);**

CE **RDR_SetDebugOutputState (char state);**

Parameter	Description
hReader	Handle of a reader to activate/deactivate broadcasting.
state	0 ... deactivate debug output 1 ... activate debug output.

1.3.30 RDR_SetReaderConfig

The **RDR_SetReaderConfig** function sets and applies the reader with specified configuration. The communication device even adapted to this new settings.

Win **RDR_SetReaderConfig(void* hReader, struct readerConfig* config);**

CE **RDR_SetReaderConfig(struct readerConfig* config);**

Parameter	Description
hReader	Handle of reader to set configuration.
config	Pointer to structure of the configuration data. For structure definition of reader configuration see header files.

Structure Definition

```
struct readerConfig {
```

```
    long baudRate;
```

```
    char protocol;
```

```
    unsigned char stationID; };
```

- **baudRate**: Defines the baud rate of the reader.
Valid values are 9600, 19200, 38400, 57600, 115200, 230400 and 460800.
- **protocol**: Defines the protocol of the reader.
Zero means ASCII protocol and one means binary protocol.
- **stationID**: Defines the station ID of the reader. Valid range is 1-254.

Remarks

This function does not work if unknown reader type is used.

Use this function for only one reader at the same time. Otherwise the communication to other devices will fail.

Be sure the used interface supports the baud rate.

1.3.31 RDR_StartTimer

The **RDR_StartTimer** function starts the timer to measure the response timing of a reader specified with its handle.

Win **RDR_StartTimer (void* hReader);**

CE **RDR_StartTimer ();**

Parameter	Description
hReader	Handle of a reader to activate/deactivate broadcasting.

Remarks

Call the **RDR_GetTiming** function to get the response timing.

1.3.32 RDR_UpdateFirmware

The **RDR_UpdateFirmware** function updates one line of the firmware. Call this function until the upload is finished.

Win **char RDR_UpdateFirmware (void* hReader);**

CE **char RDR_UpdateFirmware ();**

Parameter	Description
hReader	Handle of reader to set configuration.

Return value

If the upload is complete, the return value is zero. The reader needs an additionally reset command.

If the upload of one line is successfully completed, the return value is -1. Call this function again to upload the next line.

Error Code	Description
1	Error: Unexpected end of data
2	Error: Error during update process.
3	Error: Frame error
4	Error: Firmware type does not match with module type
5	Error: Reader does not respond
6	Error: No bootloader active
7	Error: Invalid reader handle
8	Error: Unknown error

Remarks

Before uploading the first line call **RDR_InitUpdateFirmware**.

1.4 Windows CE specific functions

1.4.1 RDR_GetHighBaudrates

The function returns the activation state of higher baud rates (230400 and 460800 baud).

```
char RDR_GetHighBaudrates();
```

Return value

If the high baud rates are active, the return value is one.

If the high baud rates are deactivated, the return value is zero.

1.4.2 RDR_GetResumeState

The function is implemented to check the power state of a handheld device. The resume state is entered after the device is power up. The reader device only works correctly if the **RDR_GetResumeState** function returns 0.

```
char RDR_GetResumeState();
```

Return value

If the reader device operates correctly, the return value is zero.

If the reader device is resuming after power up, the return value is one.

If resuming fails, the return value is -1.

1.4.3 RDR_SetHighBaudrates

The function activates or deactivates the use of high baud rates (230400 and 460800 baud).

```
void RDR_SetHighBaudrates(char state);
```

Parameter	Description
state	0 ... deactivate high baud rates 1 ... activate high baud rates

1.5 Other functions

1.5.1 RDR_DESDecrypt

The **RDR_DESDecrypt** function decrypts data with DES algorithm.

Win **char* RDR_DESDecrypt(char options, char* key, char* data, long length, char* buffer);**

CE **char* RDR_DESDecrypt(char options, char* key, char* data, long length, char* buffer);**

Parameter	Description
options	Options for decryption
key	Key for decryption
data	Data to be decrypted
length	Length of data
buffer	A buffer of the decrypted data.

Return value

The function returns the buffer of decrypted data.

Options

Bit 3 - 7	Bit 2	Bit 1	Bit 0
RFU	use triple DES	RFU	disable CBC mode

1.5.2 RDR_DESEncrypt

The **RDR_DESEncrypt** function encrypts data with DES algorithm.

Win **char* RDR_DESEncrypt(char options, char* key, char* data, long length, char* buffer);**

CE **char* RDR_DESEncrypt(char options, char* key, char* data, long length, char* buffer);**

Parameter	Description
options	Options for encryption
key	Key for encryption
data	Data to be encrypted
length	Length of data
buffer	A buffer of the encrypted data.

Return value

The function returns the buffer of encrypted data.

Options

Bit 3 - 7	Bit 2	Bit 1	Bit 0
RFU	use triple DES	enable DESFire encryption mode	disable CBC mode

DESFire encryption mode

Encryption will be done as defined in the functional specification of DESFire tag.

1.5.3 RDR_DESFire

The **RDR_DESFire** function provides the command set of the DESFire tag and the DESFire SAM.

Win `char* RDR_DESFire (void* hReader, char command, char* data, char* buffer);`

CE `char* RDR_DESFire (char command, char* data, char* buffer);`

Parameter	Description
hReader	Handle of reader to send a command.
command	command code
data	ASCII: 0 terminated string containing the data
	BIN: length of data (1 byte) + data
buffer	A buffer of the return value. ASCII protocol: The buffer size has the range of 1026 bytes. BIN protocol: The buffer size has the range of 512 bytes.

Return value

In ASCII protocol mode the function returns a 0 terminated string containing received data.

In binary protocol mode the function returns a buffer of received data including a leading length byte.

Remarks

Refer to chapter "DESFire command set" for additional information.

1.5.4 RDR_GetDESFireSAMTimeout

The **RDR_GetDESFireSAMTimeout** function returns the DESFire SAM timeout used within the "DESFire command set".

Win `char RDR_GetDESFireSAMTimeout ();`

CE `char RDR_GetDESFireSAMTimeout ();`

Return value

Returns the DESFire SAM timeout, one time slice is around 9.6ms.

1.5.5 RDR_GetDLLVersion (obsolete)

The **RDR_GetDLLVersion** function returns the version of the DLL.

Win `long RDR_GetDLLVersion(void);`

CE `long RDR_GetDLLVersion(void);`

Return value

The version is returned.

Remarks

The value 100 represents version 1.00.

This function is obsolete, use **RDR_GetDLLVersionStr**.

1.5.6 RDR_GetDLLVersionStr

The **RDR_GetDLLVersionStr** function returns the version of the DLL.

The device buffer size has the range of 100 bytes

Win `char* RDR_GetDLLVersionStr(char* buffer);`

CE `char* RDR_GetDLLVersionStr(char* buffer);`

Return value

The version as 0 terminated string is returned.

1.5.7 RDR_SetDESFireSAMTimeout

The **RDR_SetDESFireSAMTimeout** function sets the DESFire SAM timeout used within the "DESFire command set".

Win `void RDR_GetDESFireSAMTimeout (char timeout);`

CE `void RDR_GetDESFireSAMTimeout (char timeout);`

Parameter	Description
timeout	Sets the DESFire SAM timeout. On time slice is around 9.6ms.

2 DESFire command set

2.1 Overview

Command Code	Description
Common commands	
00h	SAM ON/OFF
DESFire command set	
01h	Authenticate
02h	Change Key Settings
03h	Get Key Settings
04h	Change Key
05h	Get Key Version
06h	Create Application
07h	Delete Application
08h	Get Application IDs
09h	Select Application
0Ah	Format PICC
0Bh	Get Version
0Ch	Get File IDs
0Dh	Get File Settings
0Eh	Change File Settings
0Fh	Create Standard Data File
10h	Create Backup Data File
11h	Create Value File
12h	Create Linear Record File
13h	Create Cyclic Record File
14h	Delete File
15h	Read Data
16h	Read Records
17h	Write Data
18h	Write Records
19h	Get Value
1Ah	Credit
1Bh	Debit
1Ch	Limited Credit
1Dh	Clear Record File
1Eh	Commit Transaction
1Fh	Abort Transaction

Command Code	Description
20h	Set File Settings
2Fh	Init PCB Block Number
SAM command set	
30h	Disable Crypto
31h	Change Key Entry
32h	Get Key Entry
33h	Change KUC Entry
34h	Get KUC Entry
35h	Change Key PICC
36h	Get Version
37h	Authenticate Host
38h	Select Application
39h	Dump Session Key
3Ah	Load Init Vector
3Bh	Authenticate PICC 1
3Ch	Authenticate PICC 2
3Dh	Verify MAC
3Eh	Generate MAC
3Fh	Decipher Data
40h	Encipher Data
42h	Set Logical Channel
43h	Set Transmission Factor

2.2 Error Codes

2.2.1 Common Error Codes

Error Code	Description
'?'	Unknown command
'C'	CRC or MAC verify error
'F'	Common error
'N'	No tag in field
'O'	Operation mode failure
'R'	Parameter out of range
'X'	Access denied

2.2.2 DESFire Error Codes

Error Code	Description
00h	Command successful executed
'D' xxh	DESFire error code
'S' xxyyh	SAM error code

2.2.3 SAM Error Codes

Error Code	Description
xxxxh	ISO 7816 error code

2.3 Commands

2.3.1 00h: SAM ON/OFF

This command switches on/off the SAM.

Parameter	Description
Mode	SAM Mode (1 byte)

Return Value	Description
Data	SAM Mode (1 byte)
Error	Refer to error code chapter or DESFire documentation.

SAM Mode

Mode	Description
00h	Switches OFF SAM support.
01h	Switches ON SAM support and use SAM for encryption and decryption.
02h	Switches ON SAM support: SAM is only used for key management. Encryption and decryption is done within the DLL.

Remarks

SAM Mode 02h needs the session key, so the used key has to grant dumping the session key.

Before removing the SAM from the socket the SAM has to be turned off, otherwise the SAM may be damaged.

2.3.2 01h: Authenticate

Authenticate to a DESFire card. Authentication depends on access conditions of an application or file.

SAM Mode 0

Parameter	Description
KeyNoTag	Key number on tag (1 byte)
Key	Used key (16 bytes)

SAM Mode 1 and 2

Parameter	Description
KeyNoTag	Key number on tag (1 byte)
AuthMode	Authentication mode (1 byte)
KeyNoSAM	Key number on SAM (1 byte)
KeyVersion	Key version of key number on SAM (1 byte)
TagUID	(optional) Tag UID for key diversification declared in AuthMode (7 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.3 02h: Change Key Settings

This command changes the key settings of the selected application.

Parameter	Description
KeySettings	New key settings (1 byte)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.4 03h: Get Key Settings

This command returns the key settings of the selected application.

Return Value	Description
Data	00h Positive acknowledge (1 byte). Key settings (2 bytes).
Error	Refer to error code chapter or DESFire documentation.

2.3.5 04h: Change Key

This command changes the key of the selected application.

ChangeKeyPICC command of SAM not used

Parameter	Description
KeyNoTag	Key number on tag (1 byte)
KeySettings	Key settings
OldKey	Old key (16 bytes)
NewKey	New key (16 bytes)

ChangeKeyPICC command of SAM used

Parameter	Description
KeyNoTag	Key number on tag (1 byte)
KeyCompMeth	Key compilation method (1 byte)
OldKeyNoTag	Old key number entry on SAM (1 byte)
OldKeyVersion	Old key version of old key entry on SAM (1 byte)
NewKeyNo	New key number entry on SAM (1 byte)
NewKeyVersion	New key version of new key entry on SAM (1 byte)
TagUID	(optional) Tag UID for key diversification declared in KeyCompMeth (7 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.6 05h: Get Key Version

This command reads the key version of a key of the selected application.

Parameter	Description
KeyNoTag	Key number on tag (1 byte)

Return Value	Description
Data	00h Positive acknowledge (1 byte). Key version (1 byte).
Error	Refer to error code chapter or DESFire documentation.

2.3.7 06h: Create Application

This command creates an application on a tag. Applications can only be created in the master application (000000h).

Parameter	Description
AID	Application ID (3 bytes)
KeySettings	Key settings (1 byte)
NumKeys	Number of keys (1 byte)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.8 07h: Delete Application

This command deletes an application on a tag.

Parameter	Description
AID	Application ID (3 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.9 08h: Get Application IDs

This command returns all application IDs on a tag.

Return Value	Description
Data	00h Positive acknowledge (1 byte) Application ID (each 3 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.10 09h: Select Application

This command selects a specific application for further access. An application must be selected to access all files stored in it.

Parameter	Description
AID	Application ID (3 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.11 0Ah: Format PICC

This command formats the tag. All applications are deleted. The format command requires a successful authentication.

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.12 0Bh: Get Version

This command returns the manufacture data of a tag.

Return Value	Description
Data	00h Positive acknowledge (1 byte) Manufacture data (28 bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

For more detailed information on manufacture data refer to DESFire documentation.

2.3.13 0Ch: Get File IDs

This command returns all found File IDs of the selected application.

Return Value	Description
Data	00h Positive acknowledge (1 byte) File ID (each 1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.14 0Dh: Get File Settings

This command returns additional information of a file.

Parameter	Description
FileNo	File number (1 bytes)

Return Value	Description
Data	00h Positive acknowledge (1 byte) File settings (7/17/13 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.15 0Eh: Change File Settings

This command changes the access rights of a file.

Parameter	Description
NewComSet	New communication settings (1 bytes)
NewAccess	New access rights (2 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.16 0Fh: Create Standard Data File

This command creates a standard data file in the selected application.

Parameter	Description
FileSize	File size (3 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.17 10h: Create Backup Data File

This command creates a backup data file in the selected application. Backup data files use a shadow register for data manipulation operations. The file number must be in the range from 00h to 07h.

Parameter	Description
FileSize	File size (3 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.18 11h: Create Value File

This command creates a value file in the selected application. Value blocks are signed long numbers, which are stored in Intel format (LSB first). Value files use a shadow register for data manipulation operations. The file number must be in the range of 00h to 07h.

Parameter	Description
LimitLow	Lower limit (4 bytes)
LimitUp	Upper limit (4 bytes)
Value	Value (4 bytes)
LimitedCredit	Enable limited credit (1 byte)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.19 12h: Create Linear Record File

This command creates a linear record file in the selected application. Linear record files use a shadow register for data manipulation operations. The file number must be in the range of 00h to 07h.

Parameter	Description
RecSize	Record size (3 bytes)
NumRec	Number of records (3 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.20 13h: Create Cyclic Record File

This command creates a cyclic record file in the selected application. Cyclic record files use a shadow register for data manipulation operations. The file number must be in the range of 00h to 07h.

Parameter	Description
RecSize	Record size (3 bytes)
NumRec	Number of records (3 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.21 14h: Delete File

This command deletes a file on tag in the selected application.

Parameter	Description
FileNo	File number (1 byte)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.22 15h: Read Data

This command reads data of a data file in the selected application.

Parameter	Description
Offset	Offset (3 bytes)
Length	Number of bytes to read (3 bytes)

Return Value	Description
Data	00h Positive acknowledge (1 byte) Data (n bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

Only 230 bytes can be read at once.

If performance is important, 230 bytes should be used for plain data and 224 should be used for enciphered and MAC data, because in this cases the smaller frames from the DESFire tag are filled up completely.

The offset defines the start address of the reading. The length specifies the number of bytes, which are read. The offset and length must not exceed the limits of the file.

2.3.23 16h: Read Records

This command reads records of a record file in the selected application.

Parameter	Description
RecSize	Record size (3 bytes)
Offset	Offset (3 bytes)
Length	Number of bytes to read (3 bytes)

Return Value	Description
Data	00h Positive acknowledge (1 byte) Data (n bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

Only 230 bytes can be read at once.

If performance is important, 230 bytes should be used for plain data and 224 should be used for enciphered and MAC data, because in this cases the smaller frames from the DESFire tag are filled up completely.

One record is always read at the same time. The offset points to the record within the record file from which the reading starts. The length defines the amount of records, which have to be read. The response starts always with the oldest record. Offset 000000h points to the latest record. Length 00h reads all records of the record file.

2.3.24 17h: Write Data

This command writes data to a data file in the selected application. The write command for backup data file must be validated with the commit/abort transaction command.

Parameter	Description
Offset	Offset (3 bytes)
Data	Data (n bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

Only 230 bytes can be written at once.

If performance is important, 230 bytes should be used for plain data and 224 should be used for enciphered and MAC data, because in this cases the smaller frames from the DESFire tag are filled up completely.

2.3.25 18h: Write Records

This command writes records to a data file in the selected application. A write command will append a new record to the record file until all records are filled up in a linear record file. Using cyclic record files, the oldest record is updated when all records are used up. The write command must be validated with the commit/abort transaction command.

Parameter	Description
Offset	Offset (3 bytes)
Records	Number of records (3 bytes)
Data	Data (n bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

Only 230 bytes can be written at once.

If performance is important, 230 bytes should be used for plain data and 224 should be used for enciphered and MAC data, because in this cases the smaller frames from the DESFire tag are filled up completely.

2.3.26 19h: Get Value

This command reads a value block of a value file.

Return Value	Description
Data	00h Positive acknowledge (1 byte) Value (4 bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.27 1Ah: Credit

This command increases a value in a value file. All value manipulation commands are accumulated in a shadow register. This shadow register is only be written after a successful commit transaction command.

Parameter	Description
Data	Data (4 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.28 1Bh: Debit

This command decreases a value in a value file. All value manipulation commands are accumulated in a shadow register. This shadow register is only be written after a successful commit transaction command.

Parameter	Description
Data	Data (4 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.29 1Ch: Limited Credit

This command limits the credit in a value file. The value depends on all previous debit values. All value manipulation commands are accumulated in a shadow register. This shadow register is only be written after a successful commit transaction command.

Parameter	Description
Data	Data (4 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

Remarks

This command needs the "Set File Settings" command before.

2.3.30 1Dh: Clear Record File

This command clears the whole content of a record file. After a commit transaction command the changes are written.

Parameter	Description
FileNo	File number (1 byte)

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.31 1Eh: Commit Transaction

This command validates all previous write or operations to a backup data or record files and data manipulations on value files in the selected application. All changes are done at the same time.

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.32 1Fh: Abort Transaction

This command aborts all previous write or operations to a backup data or record files and data manipulations on value files in the selected application. No data are changed. Power loss will be interpreted as an abort transaction command.

Return Value	Description
00h	Positive acknowledge (1 byte)
Error	Refer to error code chapter or DESFire documentation.

2.3.33 20h: Set File Settings

This command prepares other commands with necessary file settings.

Parameter	Description
FileNo	File number (1 byte)
ComSet	Communication settings (1 byte)
Access	Access rights (2 bytes)

Return Value	Description
00h	Positive acknowledge (1 byte)

Remarks

Following commands need a "Set File Settings" command before: Change File Settings, Create Standard Data File, Create Backup Data File, Create Value File, Create Linear Record File, Create Cyclic Record File, Read Data, Read Records, Write Data, Write Records, Get Value, Credit, Debit, Limited Credit.

2.3.34 2Fh: Init PCB Block Number

This command initializes the block number of the PCB byte. This needs to be done every time a tag was selected.

Parameter	Description
None	

Return Value	Description
00h	Positive acknowledge (1 byte)

Remarks

If the block number is not initialized after selecting a card the communication does not work. The new DESFire 8 needs an initialized block number, otherwise this tag will not work properly.

2.3.35 30h: Disable Crypto

This command allows to permanently and irreversibly disable cryptographic functionality of the SAM.

Parameter	Description
ProMas	Programming Mask (2 bytes)

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.36 31h: Change Key Entry

This command updates any key entry stored in the DESFire SAM.

Parameter	Description
OldKeyNoCEK	Old current key entry (1 byte)
KeyNo	Number of key entry to be updated (1 byte)
ProMas	Programming mask (1 byte)
NewKeyEntry	New key entry (57 bytes)
SamUID	(optional) Sam UID for key diversification (7 bytes)

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.37 32h: Get Key Entry

This command allows reading the key entry specified in the parameter KeyNo.

Parameter	Description
KeyNo	Number of key entry to be read (1 byte)

Return Value	Description
Data	Key entry (12 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.38 33h: Change KUC Entry

This command updates any key usage counter stored in the DESFire SAM.

Parameter	Description
KeyNoCLim	Logged in with current key entry (1 byte)
RefNoKUC	Number of KUC entry to be updated (1 byte)
ProMas	Programming mask (1 byte)
NewKUCEntry	New KUC entry (6 bytes)
SamUID	(optional) Sam UID for key diversification (7 bytes)

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.39 34h: Get KUC Entry

This command allows reading out the key usage counter entry specified within the parameter RefNoKUC.

Parameter	Description
RefNoKUC	Number of KUC entry to be read (1 byte)

Return Value	Description
Data	KUC entry (10 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.40 35h: Change Key PICC

This command generates a cryptogram which has to be sent to the PICC in order to change any key stored in the PICC. Both the current and the new key need to be stored in the SAM to execute this command.

Parameter	Description
KeyCompMeth	Key compilation method (1 byte)
OldKeyNo	Old key entry number (1 byte)
OldKeyVersion	Old key version of old key entry number (1 byte)
NewKeyNo	New key entry number (1 byte)
NewKeyVersion	New key version of old key entry number (1 byte)
SamUID	(optional) Sam UID for key diversification (7 bytes)

Return Value	Description
Data	Key cryptogram (24 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.41 36h: Get Version

This command returns manufacturing related data of the SAM.

Return Value	Description
Data	Manufacturing related data of the SAM (28 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.42 37h: Authenticate Host

This command authenticates to the SAM.

Parameter	Description
AuthMode	Authentication mode (1 byte)
KeyNo	Key entry number (1 byte)
KeyVersion	Key version of key entry number (1 byte)
Key	New key (16 byte)
SamUID	(optional) Sam UID for key diversification (7 bytes)

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.43 38h: Select Application

This command is the equivalent of the Select Application command of DESFire. The SAM generates a list of keys linked to the specified application ID as defined in the key entries.

Parameter	Description
AID	Application ID (3 bytes)

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.44 39h: Dump Session Key

This command can be used to retrieve the session key generated by the SAM.

Return Value	Description
Data	Session key (16 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.45 3Ah: Load Init Vector

This command is used to load an Init Vector to the 3DES hardware of the DESFire SAM.

Parameter	Description
InitVector	Init vector (8 bytes)

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.46 3Bh: Authenticate PICC 1

This command is the first part of authentication to DESFire tag.

Parameter	Description
AuthMode	Authentication mode (1 byte)
KeyNo	Key number (1 byte)
KeyVersion	Key version (1 byte)
EncRndB	Encrypted random number B, as received from the DESFire tag (8 byte)
TagUID	(optional) Tag UID for key diversification (7 bytes)

Return Value	Description
Data	Encrypted random numbers A and B, ready to send to the DESFire tag (16 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.47 3Ch: Authenticate PICC 2

This command is the second part of authentication to DESFire tag.

Parameter	Description
EncRndA	Encrypted random number A, as received from the DESFire tag (8 byte)
Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

2.3.48 3Dh: Verify MAC

This command verifies the MAC which was sent by the DESFire tag based on the given MAC plain text data and the currently valid cryptographic key.

Parameter	Description
n Bytes	Data
4 Bytes	MAC

Return Value	Description
9000h	Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

Only 230 data bytes can be sent at once.

2.3.49 3Eh: Generate MAC

This command creates a MAC which is meant to be sent to the DESFire tag based on the given plain text data and the current valid cryptographic key.

Parameter	Description
n Bytes	Data

Return Value	Description
Data	MAC (4 bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

Only 230 data bytes can be sent at once.

2.3.50 3Fh: Decipher Data

This command deciphers data packages sent by DESFire tag based on the currently valid cryptographic key and returns plain data.

Parameter	Description
3 Bytes	Read length
n Bytes	Deciphered data (8-230 bytes)

Return Value	Description
Data	Plain data (n bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

Only 230 data bytes can be sent at once.

2.3.51 40h: Encipher Data

This command creates data packages which are meant to be sent to the DESFire tag based on the given plain text and the currently valid cryptographic key.

Parameter	Description
n Bytes	Plain data (1-230 bytes)

Return Value	Description
Data	Enciphered data (n bytes) 9000h Positive acknowledge (2 bytes)
Error	Refer to error code chapter or DESFire documentation.

Remarks

Only 230 data bytes can be sent at once.

2.3.52 42h: Set Logical Channel

This command prepares the whole command set for DESFire SAM logical channels.

Parameter	Description
1 Bytes	Logical channel
Return Value	Description
Data	Logical channel (1 byte)

2.3.53 43h: Set Transmission factor

This command overwrites the default transmission factor for the DESFire SAM communication.

If the speed is set to zero, the default transmission factors are used.

Be sure a supported transmission factor is used.

Parameter	Description
1 Bytes	Transmission Factor
Return Value	Description
Data	Transmission factor (1 byte)

3 Script language

3.1 Overview

antenna on	Switch on antenna field
antenna off	Switch off antenna field
continuous read	Continuous read all tags
copy vblock	Copy value block
dec vblock	Decrement value block
highspeed select	Highspeed select
inc vblock	Increment value block
iso apdu	ISO 7816 APDU
iso attrib	ISO 14443-3 Attrib
iso hlta	ISO 14443-3 Halt A
iso hltb	ISO 14443-3 Halt B
iso pps	ISO 14443-4 PPS
iso rats	ISO 14443-4 RATS
iso reqa	ISO 14443-3 Request A
iso reqb	ISO 14443-3 Request B
iso sel1	ISO 14443-3 A Anti-collision/Select for Cascade Level 1
iso sel2	ISO 14443-3 A Anti-collision/Select for Cascade Level 2
iso sel3	ISO 14443-3 A Anti-collision/Select for Cascade Level 3
iso slotmarker	ISO 14443-3 B Slot Marker
iso wupa	ISO 14443-3 Wake Up A
iso wupb	ISO 14443-3 Wake Up B
led green	Switch on green LED
led off	Switch off LED
led red	Switch on red LED
lock block	Lock block permanently
login	Login
multilist	Multi tag list
multiselect	Multi tag select
read block	Read block
read register	Read EEPROM register
read userport	Read 1 bit user port
read vblock	Read value block
reset	Reset
select	Select one tag
transfer data	Transfer data telegram

use all	Use all tags
use felica	Use only Felica tags
use icode	Use only ICode® tags
use icode epc	Use only ICode® EPC tags
use icode uid	Use only ICode® UID tags
use iso14443a	Use only ISO14443A tags
use iso14443b	Use only ISO14443B tags
use iso15693	Use only ISO15693 tags
use sr176	Use only SR176 tags
use tagit	Use only Tagit® tags
version	Get version
write block	Write block
write masterkey	Write masterkey
write register	Write EEPROM register
write userport	Write 1 bit user port
write vblock	Write value block

3.2 Feature matrix

All readers do not support all script commands.

The unknown reader type does not support any script command.

script command	Mifare 1.0	all Bootloader 0.x / 1.x	all Dual 2.x	All MultiISO 1.x
antenna on	X		X	X
antenna off	X		X	X
continuous read	X		X	X
copy vblock	X		X	X
dec vblock	X		X	X
highspeed select			X	X
inc vblock	X		X	X
iso apdu	X		X	X
iso attrib			X	X
iso h1ta	X		X	X
iso h1tb			X	X
iso pps			X	X
iso rats	X		X	X
iso reqa	X		X	X
iso reqb			X	X
iso sel1	X		X	X
iso sel2	X		X	X
iso sel3	X		X	X
iso slotmarker			X	X
iso wupa	X		X	X

script command	Mifare 1.0	all Bootloader 0.x / 1.x	all Dual 2.x	All MultilSO 1.x
iso wupb			X	X
led green			X	X
led off			X	X
led red			X	X
lock block			X	X
login	X		X	X
multilist	X		X	X
multiselect	X		X	X
read block	X		X	X
read register	X		X	X
read userport	X		X	X
read vblock	X			X
reset	X	X	X	X
select	X		X	X
transfer data	X		X	X
use all			X	X
use icode				X
use icode epc				X
use icode uid				X
use iso14443a			X	X
use iso14443b			X	X
use iso15693				X
use sr176				X
use tagit				
version	X	X	X	X

script command	Mifare 1.0	all Bootloader 0.x / 1.x	all Dual 2.x	All MultilSO 1.x
write block	X		X	X
write masterkey	X		X	
write register	X		X	X
write userport	X		X	X
write vblock	X		X	X

3.3 Commands

3.3.1 antenna on

This command switches on the antenna field.

Command	Data
antenna on	None

Answer	Description
'P'	Positive acknowledge

Example

Command	Parameters	Description
antenna on	-	Switch on the antenna field.

3.3.2 antenna off

This command switches off the antenna field and it also reduces the power consumption.

If a command is sent the antenna field is switched on automatically.

A power down will reset all tags in the field. A new selection, login is needed.

Command	Data
antenna off	None

Answer	Description
'P'	Positive acknowledge

Example

Command	Parameters	Description
antenna off	-	Switch off the antenna field.

3.3.3 continuous read

The serial numbers are repeated continuously while one or more tags remain in the field. This command stops if any character is sent to the reader module.

Command	Data
continuous read	None

Answer	Description
Data	Serial number(s) of tag(s) in the field. Data length depends on type of tag.
None	If no tags are in the field no answer is returned.
Error	For error codes refer to reader documentation (1 byte).

Remarks

This command is not supported in the binary protocol mode.

3.3.4 copy vblock

This command copies a value block to another block.

Command	Data
copy vblock	Source block address (1 byte) Destination block Address (1 byte)

Answer	Description
Data	New value block data of destination block (4 bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
copy vblock	0405	Copies value block 04 to value block 05.

Remarks

See tag specification for correct access to a value block.

3.3.5 dec vblock

This command subtracts a value from a value block.

Command	Data
dec vblock	Block Address (1 byte) Value (n bytes)

Answer	Description
Data	New value (4 bytes)
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
dec vblock	0400000064	Subtracts the value 100 from block 04.

Remarks

See tag specification for correct access to a value block.

The byte length of the value depends on the used tag.

3.3.6 highspeed select

This command selects a single card in the antenna field similar to the select command, switches to high baud rates and enables 256 byte frames.

Command	Data
highspeed select	Baud rate (1 byte)

Answer	Description
Data	Serial number (n bytes) Used frame size and baud rate (1 byte).
Error	For error codes refer to reader documentation (1 byte).

3.3.7 inc vblock

This command adds a value to a value block.

Command	Data
inc vblock	Block address (1 byte) Value (n bytes)

Answer	Description
Data	New Value (4 bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
inc vblock	0400000064	Adds value 100 to block 04.

Remarks

See tag specification for correct access to a value block.

The byte length of the value depends on the used tag.

3.3.8 iso apdu

This command sends the ISO 7816 APDU command.

Command	Data
iso apdu	PCB (1 byte) (optional) CID (1 byte) (optional) NAD (1 byte) ISO 7816 APDU (n bytes)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

If no answer is returned, adjust the ReaderDLL timeout with **SetCommTimeout**.

Some APDU commands may need several seconds and the default timeout of the ReaderDLL is set to 300ms.

For PCB, CID, NAD and ISO 7816 APDU refer to ISO 14443-4 and ISO 7816.

3.3.9 iso attrib

This command sends the ISO 14443-3 B Attrib command.

Command	Data
iso attrib	UID (4 bytes) Param 1 (1 byte) Param 2 (1 byte) Param 3 (1 byte) Param 4 (1 byte) Higher layer – INF (n bytes)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.10 iso hlta

This command sends the ISO 14443-3 A Halt command.

Command	Data
iso hlta	None

Answer	Description
00h	Positive acknowledge (1 byte).
Error	For error codes refer to reader documentation (1 byte).

Remarks

This command does not have any response.

The answer depends on used reader.

3.3.11 iso hltb

This command sends the ISO 14443-3 B Halt command.

Command	Data
iso hltb	UID (4 bytes)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.12 iso pps

This command sends the ISO 14443-4 PPS command.

Command	Data
iso pps	CID (1 byte) (optional) DSI/DIR (1 byte)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

Coding of DSI/DRI refer to ISO 14443-4.

3.3.13 iso rats

This command sends the ISO 14443-4 RATS command.

Command	Data
iso rats	CID (1 byte)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

Be sure not to send an additionally command after the RATS command until the SFGT guard timeout is elapsed.

3.3.14 iso reqa

This command sends the ISO 14443-3 A Request command.

Command	Data
iso reqa	None

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.15 iso reqb

This command sends the ISO 14443-3 B Request command.

Command	Data
iso reqb	None

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.16 iso sel1

This command sends the ISO 14443-3 A Anti-collision or Select command for cascade level 1.

Command	Data
iso sel1	NVB (Number of valid bits) (1 byte) UID (4 bytes)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

If NVB is 70h, the select command is used and a BCC byte has to be added, otherwise the anti-collision command is used.

Coding of NVB refer to ISO 14443-3.

3.3.17 iso sel2

This command sends the ISO 14443-3 A Anti-collision or Select command for cascade level 2.

Command	Data
iso sel2	NVB (Number of valid bits) (1 byte) UID (4 bytes)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

If NVB is 70h, the select command is used and a BCC byte has to be added, otherwise the Anti-collision command is used.

Coding of NVB refer to ISO 14443-3.

3.3.18 iso sel3

This command sends the ISO 14443-3 A Anti-collision or Select command for cascade level 3.

Command	Data
iso sel3	NVB (Number of valid bits) (1 byte) UID (4 bytes)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

If NVB is 70h, the select command is used and a BCC byte has to be added, otherwise the Anti-collision command is used.

Coding of NVB refer to ISO 14443-3.

3.3.19 iso slotmarker

This command sends the ISO 14443-3 B Slot Marker command.

Command	Data
iso slotmarker	Slot Number, valid range 1 – 15 (1 byte)

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.20 iso wupa

This command sends the ISO 14443-3 A Wake Up command.

Command	Data
iso wupa	None

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.21 iso wupb

This command sends the ISO 14443-3 B Wake Up command.

Command	Data
iso wupb	None

Answer	Description
Data	Number of received data bytes (1 byte). Data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For more detailed information refer to ISO 14443-3.

3.3.22 led green

This command switches on the green LED.

Command	Data
led green	None

Answer	Description
'DG'	Positive acknowledge (2 bytes).

3.3.23 led off

This command switches off the LED.

Command	Data
led off	None

Answer	Description
'DN'	Positive acknowledge (2 bytes).

3.3.24 led red

This command switches on the red LED.

Command	Data
led red	None

Answer	Description
'DR'	Positive acknowledge (2 bytes).

3.3.25 lock block

This command locks a block permanently.

Command	Data
lock block	Block address (1 byte)

Answer	Description
Data	Answer depends on used reader (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
lock block	01	Locks block 01 permanently.

Remarks

This command is irreversible.

3.3.26 login

The login command performs an authentication to one sector of a card. Only one sector can be accessed at the same time. The login command needs a valid key pair to succeed. The tag has to be selected previously.

Mode 1

Command	Data
login	Sector (1 byte) Carriage return (1 byte)

Mode 2

Command	Data
login	Sector (1 byte) Key type (1 byte)

Mode 3

Command	Data
login	Sector (1 byte) Key type (1 byte) Carriage return (1 byte)

Mode 4

Command	Data
login	Sector (1 byte) Key type (1 byte) Key (6 bytes)

Answer	Description
'L'	Positive acknowledge (1 byte).
Error	For error codes refer to reader documentation (1 byte).

Remarks

For the key type refer to reader documentation.

Example

Command	Data	Description
login	01<CR>	Login in sector 01h, using transport key A.
login	0732	Login in sector 07h, using EEPROM key 2. key type B.
login	02AA<CR>	Login in sector 02h, using transport key A.
login	0FAA001122334455	Login in sector 0Fh, using key 001122334455h, key type A

3.3.27 multilist

This command detects all tags in the antenna field and returns present tag list.

Command	Data
multilist	None

Answer	Description
Data	List of the serial numbers of all present tags (n bytes). Number of tags found (1 byte).
Error	For error codes refer to reader documentation (1 byte).

3.3.28 multiselect

This command selects a tag in multi tag environment.

Command	Data
multiselect	Serial number of tag (n bytes)

Answer	Description
Data	Serial number of tag (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
multiselect	C297A898	Select tag with the serial number C297A898.

Remarks

Length of serial number depends on tag.

3.3.29 read block

This command reads a data block of a tag.

Command	Data
read block	Block address (1 byte)

Answer	Description
Data	Block data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
read block	04	Read block 04h.

Remarks

See tag specification for correct access to a block.

3.3.30 read register

This command reads a register of the reader EEPROM.

Command	Data
read register	Register address (1 byte)

Answer	Description
Data	Register data (1 byte).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
read register	0A	Read register 0Ah.

3.3.31 read userport

This command returns the state of the user port.

Command	Data
read userport	State of user port (1 byte)

Answer	Description
Data	Status of user port, 0 or 1 (1 byte).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
read userport	-	Read user port.

3.3.32 read vblock

This command reads a value block.

Command	Data
read vblock	Block address (1 byte)

Answer	Description
Data	Value (4 bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
read vblock	04	Read value block 04h.

Remarks

See tag specification for correct access to a value block.

3.3.33 reset

This command executes a power on (software) reset. Startup configuration settings will be loaded.

Command	Data
reset	None

Answer	Description
Data	Answer depends on used protocol type und used reader (n bytes).
Error	For error codes refer to reader documentation (1 byte).

3.3.34 select

This command selects a single card in the antenna field. It can only be used in single tag mode.

Command	Data
select	None

Answer	Description
Data	Serial number of selected tag (n bytes).
Error	For error codes refer to reader documentation (1 byte).

3.3.35 transfer data

This command is designed to send specific frames to a tag.

Command	Data
transfer data	Data depends on reader and used tag (n bytes)

Answer	Description
Data	Data depends on reader, used tag and send data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

3.3.36 use all

This command configures a reader to use all known tags that are supported.

Command	Data
use all	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.37 use icode

This command configures a reader to use only ICode® tags.

Command	Data
use icode	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.38 use icode epc

This command configures a reader to use only ICode® EPC tags.

Command	Data
use icode epc	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.39 use icode uid

This command configures a reader to use only ICode® UID tags.

Command	Data
use icode uid	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.40 use iso14443a

This command configures a reader to use only ISO14443A tags.

Command	Data
use iso14443a	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.41 use iso14443b

This command configures a reader to use only ISO14443B tags.

Command	Data
use iso14443b	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.42 use iso15693

This command configures a reader to use only ISO15693 tags.

Command	Data
use iso15693	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.43 use sr176

This command configures a reader to use only SR176 tags.

Command	Data
use sr176	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.44 use tagit

This command configures a reader to use only Tagit® tags.

Command	Data
use tagit	None

Answer	Description
Data	Data depends on reader (n bytes).

3.3.45 version

This command returns the version string of a reader.

Command	Data
version	None

Answer	Description
Data	Version string of a reader (n bytes).

3.3.46 write block

This command writes data to a block.

Command	Data
write block	Block address (1 byte) Block data (n bytes)

Answer	Description
Data	Block data (n bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
write block	0411223344...	Write the value 11223344...h to the block 04h.

Remarks

See tag specification for correct access to a block.

Block length depends on used tag.

3.3.47 write masterkey

This command stores a master key into the reader memory. Master keys are non-volatile.

Command	Data
write masterkey	Key number (1 byte) Key (6 bytes)

Answer	Description
Data	Written key (6 bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
write masterkey	00112233445566	Write key 112233445566h to EEPROM as key 0.

Remarks

Master keys are not readable!

3.3.48 write register

This command writes a value into a register of a reader EEPROM.

Command	Data
write register	Register address (1 byte) New register value (1 byte)

Answer	Description
Data	Written register value (1 byte).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
write register	0AFF	Write the value FFh into the register 0Ah.

3.3.49 write userport

This command sets the user port.

Command	Data
write userport	State of user port, 0 or 1 (1 byte)

Answer	Description
Data	Written state of user port, 0 or 1 (1 byte).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
write userport	01	Write 01h to user port.

3.3.50 write vblock

This command writes a value to a value block.

Command	Data
write vblock	Block address (1 byte) Block data (n bytes)

Answer	Description
Data	Written value (4 bytes).
Error	For error codes refer to reader documentation (1 byte).

Example

Command	Data	Description
write vblock	0411223344...	Write value 11223344...h to value block 04h.

Remarks

See tag specification for correct access to a value block.

Block length depends on used tag.

4 References

- [1] DESFire documentation, Philips <http://www.semiconductors.philips.com>
- [2] Data Encryption Standard (DES), FIPS PUB 46-3,
Reaffirmed 1995 October 25
- [3] International Standard ISO/IEC 14443, Part 1-4

APPENDIX A - C++ Example Listing

This example listing is tested with MS Visual C++ .NET 2003.

```
char buffer[1026], detectedReader[256];
void *hReader, *hComm;

// Initialize the communication device "com1" and use auto detection.
//
// If you do not want to use auto detection then set the communication
// parameters with the third parameter.
hComm = RDR_OpenComm("com1", 1, NULL);
if (hComm == NULL)
    printf("Error opening communication device\n");
else
{
    // Detect all available readers.
    //
    // If you know your reader ids, you do not have to call
    // DetectReader.
    RDR_DetectReader(hComm, detectedReader);

    // Open the first detected reader and auto detect reader type.
    //
    // If you know your reader type, you can specify it as third
    // parameter.
    hReader = RDR_OpenReader(hComm, (unsigned char)detectedReader[0], 0);

    if (hReader == NULL)
        printf("Can not open reader with id %i\n", detectedReader[0]);
    else
    {
        // Reader successfully opened.
        RDR_SendCommandGetData(hReader, "version", "", buffer);
        printf("Version: %s\n", buffer);

        //
        // Write some code for reader here.
        // ...
        //

        // Close the opened reader.
        RDR_CloseReader(hReader);
    }

    // Release the communication device.
    RDR_CloseComm(hComm);
}
```

APPENDIX B - C++ Sample Application

A C++ sample application with source code can be found inside the Samples directory within this software package.

The sample was designed with MS Visual C++ .NET 2003.

APPENDIX C - VB .NET without Marshal Casting

Note that VB .NET is not supported.

You can find a Version 3.0.0 Header inside the unsupported directory within this software package.

Workaround:

1. Import the System.Text into your visual basic source file.

```
Imports System.Text
```

2. Substitute all marshal castings

```
<MarshalAs(UnmanagedType.VBByRefStr)> ByRef buffer As String
```

with a byte array declaration:

```
ByVal buffer As Byte()
```

3. Declare all variable definitions like this:

```
Dim buffer(514) As Byte
```

4. Use the following functions to set the byte buffer or get a string from a byte buffer:

- a. Initialize the buffer:

```
buffer = Encoding.ASCII.GetBytes("blabla")
```

- b. Read the buffer as string:

```
text = Encoding.ASCII.GetString(buffer)
```

APPENDIX D - Static Library

The static library "Reader.lib" is compiled with Visual C++ .NET 2003 as single threaded target without debug information.

Use the header file "ReaderLib.h" to get the function prototypes.

Multithreaded targets or enabled debug information may raise the warning LNK4098.

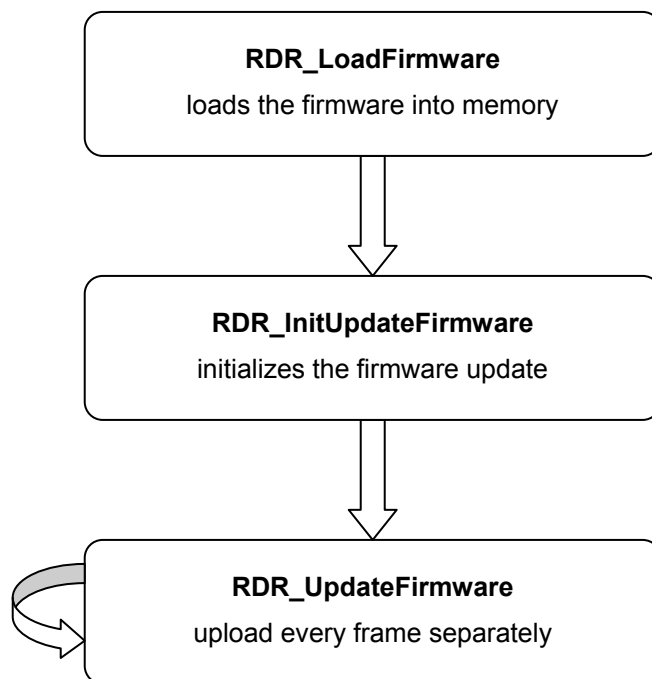
Use the linker option "/NODEFAULTLIB:LIBC.LIB" to suppress this warning.

APPENDIX E - Paypass functionality

Please note that the Paypass functionality is excluded from this documentation.

You can obtain the Paypass documentation separately.

APPENDIX F - Firmware update



APPENDIX G - Version History

November 3, 2008		Version 4.1.0, Revision 1.0
February 2008		Version 4.0.0, Revision 1.0
September 2006		Version 3.3.0, Revision 1.0
November 2005		Version 3.2.0, Revision 1.0
June 2005	Initial Release	Version 3.1.0, Revision 1.0

Version 4.1.0

Version 4.1.0 is the development of ReaderDLL version 4.0.0, incorporating two new features.

- Adding to the existing functionality contained within the function call 'RDR_OpenComm'. Previously, it has only been possible to connect to com ports within the range 'COM1' – 'COM9'. Version 4.1.0.0 removes this restriction.
- For readers that employ the Oxford Semiconductors chipset on their compact flash interface, it is possible to operate at higher baud rates: 230kBaud, 460kBaud. However, these baud rates are achieved through a baud rate multiplier, resulting in the requested baud rate differing from the actual baud rate. In order to address this anomaly, version 4.1.0.0 includes two new function calls. These function calls enable pre-division in order to ensure that the requested baud rate matches the actual baud rate.

Version 4.0.0

- Paypass functionality added
- Fixed some bugs
- DESFire 8 support for the DESFire command set
- Support of C# discontinued
- Discontinued support of older devices
- New function **RDR_GetCommDeviceHandle** added
- Supported frame size up to 512 data bytes
- Discontinued support of old function calls without "RDR_" prefix
- **RDR_DetectSerialPort** now supported for both platforms.

Version 3.3.0

- New functions added for CF version: **RDR_SetHighBaudRates** and **RDR_GetHighBaudRates**
- New functions **RDR_SleepComm** and **RDR_WakeupComm** added
- Fixed some minor bugs if no high precision timer is available on the target system
- New function added **RDR_CommExtFunction** to control RTS and DTR pin
- New function added **RDR_AbortContinuousReadEx** for use in ASCII and binary mode
- Fixed some bugs

Version 3.2.0

- New .NET wrapper DLL for CReader.DLL
- New functions **RDR_SetDESFireSAMTimeout** and **RDR_GetDESFireSAMTimeout** for DESFire SAM
- Fixed some bugs with DES En/Decryption
- Fixed some bugs within the **DESFire** command set
- Fixed some minor bugs
- Added support of MultiISO 1.x
- Speed improvements for DES En/Decryption

Version 3.1.0

- New function **DESFire** for DESFire and SAM command set added.
- New functions for DES encryption and decryption (**RDR_DESDecrypt**, **RDR_DESEncrypt**).
- Added support of Mifare 1.0.
- Dynamic and static Library with prefix function names.
- New function to open a single reader **RDR_OpenSingle**
- Power Management for Windows CE version can be deactivated

Version 3.0.0

- Support up to 460800 baud
- Support of serial ports > 9
- Added support of LFX 1.x, Multitag 1.x, Mifare 0.14f
- Removed support of older reader modules
- New functions added **AbortContinuousRead**, **GetDLLVersionStr**
- Not common baud rates can be switched off during autodetection
- Changed response of **GetDeviceID** if reader module does not support the ID
- Added more functionality to **SetCommTimeout** and **GetCommTimeout**
- Changed some timeouts to speed up autodetection
- Changed detection order of the baud rates with autodetection
- Fixed some bugs
- Support of Windows CE.
- Additional ISO script commands.
- Buffer size increased to 514 bytes for ASCII mode.
- New functions added for firmware update **LoadFirmware**, **InitUpdateFirmware** and **UpdateFirmware**
- Removed support of VB 6.0 and VB .NET.

Version 2.0.3

- Compiled static library as single threaded target.
- Fixed a bug with static library linked with a project compiled with debug information.

Version 2.0.1

- Fixed a bug in the Reader.dll with using **StartTimer** and **GetTiming**.

Version 2.0.0

- Significant speed optimizations.
- **OpenComm** supports fast detection.
- Support of 115200 baud.
- Buffer size increased to 512 bytes in ASCII mode.
- Support of binary protocol v2.
- Debug output supported.
- Support of static linked library.

Version 1.5.1

- Added support of new "Noisy Environment" Flag.
- Fixed bug with binary protocol

Version 1.5.0

- Added support of "ISO 1.x" readers.
- Added support of "Dual 2.x" readers.
- Added support of "Bootloader 1.x" readers.
- Added support of "MULTITAG 1.x" readers.

Version 1.4.0

- Added support of "Bootloader 0.x" readers.
- Now it is possible to send data without any command.

Version 1.3.0

- Fixed a bug in **SetReaderConfig**.

Version 1.2.0

- Fixed some bugs in binary protocol.
- Code optimizations.
- Added support for "Dual 1.x" readers.
- Fixed a bug with Dual 1.0 reader and the script command "multiselect".
- **OpenReader** does not cancel continuous read mode after reset.

Version 1.1.1

- Adjust timeout value in **OpenComm**.

Version 1.1.0

- Added script commands for LED.
- Added function **SendCommandGetDataTimeout**.
- Now the ReaderDLL handles all timeouts, serial and usb, by its own.

Version 1.0.1

- Fixed an infinite loop bug in **GetDataTimeout**.

Version 1.0.0

Initial Release